ИССЛЕДОВАНИЕ НЕКОТОРЫХ КЛАССИЧЕСКИХ АЛГОРИТМОВ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА

Сехпосян Арташес

Институт проблем информатики и автоматизации НАН РА, Армения

DOI: https://doi.org/10.31435/rsglobal_ws/31032019/6398

ARTICLE INFO

Received: 18 January 2019
Accepted: 22 March 2019
Published: 31 March 2019

KEYWORDS

Transport routing task,
ZMT,
traveling salesman problem,
Dijkstra algorithm,
Floyd algorithm,
NP-complete tasks.

ABSTRACT

The article is devoted to the study of some classical algorithms for transport routing problems. The article describes the existing classical transport routing algorithms, such as the Clark-Wright algorithm and its extended algorithm. The main minus of Clarke-Wright's algorithm has been revealed, which consists in the efficiency of its operation decreases as it approaches the end of the calculations, while at the beginning of the work, the solutions are relatively successful. To improve the performance of the Clark-Wright algorithm, three approaches have been proposed in its extended algorithm. Also were explored the algorithms of Mole-Jameson and Christofides, Mingozzi and Tossa, which can be used for tasks with an unspecified in advance number of vehicles, are investigated. The algorithm of the Notes, which is used for the initial processing of the problems of ZMT, is proposed. Classical improvement algorithms for MMT are investigated, in which either a single route at a time or several routes are processed. The above algorithms give more interesting results than their predecessors, and are considered optimal in terms of the use of certain resources.

Citation: Сехпосян Арташес. (2019) Issledovanie Nekotoryh Klassicheskih Algoritmov Zadachi Marshrutizacii Transporta. *World Science*. 3(43), Vol.1. doi: 10.31435/rsglobal_ws/31032019/6398

Введение. Для выживания в условиях свободного рынка компании стремятся использовать весь арсенал доступных инструментов, позволяющих получить конкурентное преимущество перед другими компаниями. Одним из таких инструментов является оптимизация накладных затрат, которая позволяет при том же объеме используемых ресурсов увеличить прибыль. Для транспортных компаний существенная оптимизация затрат может быть достигнута в том числе и за счет построения эффективных маршрутов для транспортных средств. Именно по этой причине эффективным алгоритмам решения задачи маршрутизации транспорта (ЗМТ, VRP – vehicle routing problem) уделено столь пристальное внимание со стороны исследователей.

Задача маршрутизации транспорта (ЗМТ) является обобщением известной задачи коммивояжера и, следовательно, принадлежит к классу NP-полных задач. Это означает, что для нее не найден алгоритм решения за полиномиальное время и не доказано, что такого алгоритма не существует. Каждый маршрутизатор действует по алгоритму кратчайшего пути. Кратчайший путь можно определить с помощью некоторого математического аппарата, называемого графом. Существуют наиболее эффективные алгоритмы нахождения кратчайшего пути:

·алгоритм Дейкстры (используется для нахождения оптимального маршрута между двумя вершинами);

·алгоритм Флойда (для нахождения оптимального маршрута между всеми парами вершин).

Указанные алгоритмы легко выполняются при малом количестве вершин в графе. При увеличении их количества задача поиска кратчайшего пути усложняется.

Перечислим и анализируем некоторые важные алгоритмы для решения Задачи Маршрутизации Транспорта.

1. Некоторые классические алгоритмы.

1.1. Алгоритм Кларка-Райта.

Алгоритм Кларка-Райта (Clarke and Wright) [1] – это один из самых известных алгоритмов для решения ЗМТ. Его идея основана на процессе слияния мелких маршрутов в более крупные, проводимого до тех пор, пока есть возможность уменьшить суммарную стоимость объезда. Особую роль в этом алгоритме играет понятие "сбережения" (saving) – это снижение общей стоимости решения, получаемое при объединении двух маршрутов. Рассмотрим ситуацию, когда маршрут $(0, \dots, i, 0)$ и маршрут $(0, j, \dots, 0)$ могут быть совмещены в единую последовательность $(0, \dots, i, j, \dots, 0)$. Сбережением является изменение расстояния, равное $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, если оно больше нуля, где c_{ij} – расстояние между соответствующими вершинами. Алгоритм применяется в случаях, когда количество экипажей не определено заранее и его можно вычислять в ходе работы. Его можно использовать как для симметричных задач, так и для несимметричных, но в [2] показано, что качество работы для симметричных случаев заметно ухудшается. Известны два варианта реализации: параллельный и последовательный. В обоих случаях для них требуется предварительное выполнение подготовительного этапа. Он заключается в:

1. В вычислении сбережений $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, для $i, j = 1, \dots, n$ и $i \neq j$.
2. Создании n маршрутов транспортных средств $(0, i, 0)$ для $i = 1, \dots, n$.
3. Сортировке сбережений в порядке убывания.

Рассмотрим сначала параллельный вариант алгоритма:

1. Просматриваем построенный список сбережений с его начала и выполняем следующие действия:
2. Для текущего элемента списка s_{ij} определяем, существуют ли два маршрута, один из которых содержит дугу или ребро $(0, j)$, второй — дугу или ребро $(i, 0)$, и которые могут быть соединены в один общий маршрут.
3. Если такие маршруты найдены, то выполняем их объединение, удаляя $(0, j)$ и $(i, 0)$ а затем добавляя (i, j).

Последовательный вариант можно представить следующим образом:

1. Для всех маршрутов$(0, i, \dots, j, 0)$ выполним следующие действия:
2. Проведём поиск элемента в списке сбережений s_{ki} или s_{jl}, который может быть использован для объединения текущего маршрута с некоторым, содержащим дугу (ребро) $(k, 0)$ или $(0, l)$.
3. Если элемент был найден, то выполним объединение и применим процедуру ещё раз к новому маршруту.
4. Если сбережение найти не удалось, то перейдём к следующему маршруту и продолжим работу.
5. Завершим работу, когда объединения более невозможны.

Вычислительные результаты сравнения показывают, что параллельный вариант алгоритма даёт результаты лучше, чем последовательный.

1.2. Расширения алгоритма Кларка-Райта.

Один из недостатков алгоритма Кларка-Райта заключается в том, что эффективность его работы падает по мере приближения к концу вычислений, в то время как в начале работы решения получаются относительно удачные. В [3] и в [4] предлагалось обобщение понятия сбережения с целью устранения этого недостатка. Сбережение представляется в виде $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$, где λ – параметр для учёта формы маршрута, который позволяет сделать акцент на расстояния между вершинами для соединения. В [5] показано, что $\lambda = 0{,}4$ улучшает решения, как с точки зрения суммарного расстояния, так и с точки зрения конечного количества маршрутов.

Алгоритм Кларка-Райта может оказаться неэффективным с точки зрения времени исполнения, т. к. во всех вариантах все выигрыши должны быть вычислены, сохранены и отсортированы. Различные авторы предпринимали попытки предложить расширения, уменьшающие время исполнения и использование памяти, необходимые для работы этого алгоритма.

При реализации стратегии, основанной на понятии сбережения, необходимо уделять внимание двум важным аспектам: методу нахождения элемента наибольшего сбережения в списке и требованиям к хранению данных в памяти. Здесь поиск наибольшего сбережения – это самый трудоёмкий процесс. Для решения этой задачи применяются три подхода:

1. Использование полного алгоритма сортировки. Например, сортировки Хоара.

2. Использование ограниченной сортировки при помощи построения кучи [5]. Здесь кучей называется специальное двоичное дерево, в котором сбережения расположены так, что родительский узел всегда имеет большее значение, чем дочерние. Когда происходит слияние маршрутов, то куча позволяет быстро и эффективно исключать соответствующие узлы.

3. Использование специального итеративного алгоритма для вычисления максимального сбережения [6]. В работе было показано, что $s_{ij} > \bar{s}$ всегда, когда $c_{0i} > 0.5\bar{s}$ и $c_{0j} > 0.5\bar{s}$, на основе того, что все расстояния положительные и выполняется правило треугольника в правой части определения сбережения, где $\bar{s}$ – текущее максимальное значение сбережения. Приведённое необходимое условие затем используется для эффективного нахождения максимального сбережения.

В [7] и в [8] описана ещё одна интересная модификация алгоритма, основанного на понятии сбережения. На каждом шаге сбережение s_{pq}, полученное при слиянии маршрутов p и q, вычисляется как $s_{pq} = t(S_p) + t(S_q) - t(S_p \cup S_q)$, где S_k— множество вершин маршрута k, а $t(S_k)$— длина оптимального решения ЗК, полученного на множестве S_k. Значение $t(S_k)$ может определяться разными способами. Один из них предложен на основе использования приближённой оценке длины точного решения.

Алгоритм Кларка-Райта утратил своё значение как самостоятельного подхода, т. к. предложено много более эффективных методов, но часто используется как способ нахождения начального решения для последующего улучшения.

1.3. Последовательный алгоритм вставки Моля-Джеймсона

Алгоритм Моля-Джеймсона (Mole and Jameson) может применяться для задач с неопределённым заранее количеством транспортных средств. Он использует при расширении маршрута два параметра – λ и μ.

$$\alpha(i,k,j) = c_{ik} + c_{kj} - \lambda c_{ij},$$
$$\beta(i,k,j) = \mu c_{0k} - \alpha(i,k,j),$$

Работа алгоритма может быть представлена следующим образом:

1. Если нет неиспользованных вершин, завершаем работу алгоритма. В противном случае создаём новый маршрут $(0,k,0)$, где k – любая не использованная вершина.

2. Вычисляем для любой неиспользованной вершины k стоимость возможной вставки в новый маршрут $\alpha^*(i_k,k,j_k) = \min\{\alpha(r,k,s)\}$, где r и s – любые две соседние вершины, принадлежащие последнему созданному маршруту. Вершины i_k и j_k – две вершины, соответствующие α^*. Если возможных вариантов вставки не существует, возвращаемся на шаг 1. В противном случае вершину для вставки k^* выберем такую, чтобы она давала $\beta^*(i_{k^*},k^*,j_{k^*}) = \max\{\beta(i_k,k,j_k)\}$ среди всех возможных k. Добавим k^* между вершинами i_{k^*} и j_{k^*}.

3. После добавления вершины выполним оптимизацию при помощи процедуры 3-опт [9] и вернёмся на шаг 2.

Правила вставки контролируются параметрами λ и μ. Например, если $\lambda = 1$ и $\mu = 0$, алгоритм вставит вершину, позволяющую получить минимальное расстояние. Если $\lambda = \mu = 0$, то вставляемая вершина соответствует минимальной сумме расстояний между двумя соседними вершинами. Если $\mu = \infty$ и $\lambda > 0$, то вставляется вершина, максимально удалённая от депо.

1.4. Последовательный алгоритм вставки Кристофидеса-Мингоззи-Тосса.

Кристофидес, Мингоззи и Тосс (Christofides, Mingozzi and Toth) разработали более глубокий последовательный алгоритм вставки [10]. Он также применяется для случаев с неопределённым заранее числом транспортных средств. Это двухфазовый эвристический метод, которым также можно управлять при помощи двух параметров λ и μ.

1.5. Алгоритм заметания.

Алгоритм заметания (sweep algorithm) [11] используется для первоначальной обработки задач ЗМТ. В процессе его работы наполнение кластеров определяется поворотом луча,

исходящего из депо. Затем для каждого кластера отдельно решается ЗК. В некоторых вариантах алгоритма присутствует фаза последующей оптимизации, в которой производится обмен вершинами между соседними кластерами, после чего выполняется корректировка маршрутов. Алгоритм не использует заранее заданного количества транспортных средств. Первое упоминание этого алгоритма встречается в [12] и [13], но его обычно приписывают [11].

2. Классические улучшающие алгоритмы

Улучшающие алгоритмы для ЗМТ обрабатывают либо отдельный маршрут за раз, либо несколько маршрутов. В случае для одного маршрута можно применять любые алгоритмы оптимизации для ЗК. Для второго варианта могут быть разработаны алгоритмы, которые анализируют структуру, представленную несколькими маршрутами.

2.1. Оптимизация отдельного маршрута

Большинство процедур оптимизации для ЗК могут быть описаны в терминах $\lambda-$ опт операций, которые были предложены Лином [9]. В них λ рёбер удаляются из маршрута, и λ оставшихся сегментов пересоединяются во всех возможных комбинациях. Если улучшающее соединение найдено (первое достигнутое или наиболее удачное), то изменения вносятся в маршрут. Работа останавливается на локальном минимуме, если более невозможно найти подходящие варианты замен. Проверка $\lambda-$ оптимальности решения требует времени $O(n^{\lambda})$.

Предложены несколько вариантов этого подхода. Лин и Керниган [12] представили алгоритм, в котором λ изменяется динамически в процессе поиска. В [13] описан метод *Or*-опт, заключающийся в подмене последовательностей из 3, 2 или 1 соседних вершин на последовательность из другого фрагмента этого же маршрута. Фактически, он представляет собой ограниченную форму 3-опт оптимизации. Проверка *Or*-оптимальности требует времени $O(n^2)$. На той же основе в [11] предлагается ограниченная форма 4-опт алгоритма, называемая 4-opt* и требующая для проверки оптимальности времени $O(\omega n^2)$.

В [14] проведён анализ упомянутых методов, где авторы пришли к заключению, что оптимизация Лина-Кернигана даёт в среднем самые хорошие результаты.

2.2. Алгоритмы для улучшения нескольких маршрутов.

В [15, 16, 17] приводятся алгоритмы, основанные на обмене рёбер между маршрутами. Они вобрали в себя различные схемы, предложенные ранее несколькими авторами [18, 19, 20].

Особое внимание уделяется общей схеме b-циклического k-переноса, в которой рассматривается циклическая перестановка b маршрутов, и выполняется перенос k вершин из каждого маршрута в следующий маршрут. Авторы показывают, что применение некоторых обменов b-циклического k-переноса (с $b = 2$ или с переменным значением b и $k = 1$ или $k = 2$) даёт интересные результаты.

В [15] перечислены основные варианты модификации маршрутов, они являются частными случаями 2-циклических переносов.

1. Перекрещивание двух рёбер из двух маршрутов (рис. 1).
2. Обмен вершинами между двумя маршрутами (рис. 2).
3. Перенос вершин из маршрута в маршрут (рис. 3).
4. Комбинации предыдущих вариантов.

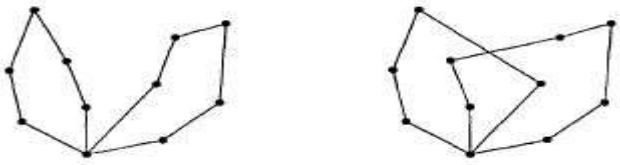

Рис. 1. Пересечение двух рёбер двух маршрутов

Рис. 2. Обмен вершинами двух маршрутов

Рис. 3. Перенос вершин из маршрута в маршрут

3. Заключение. Применение классических алгоритмов позволяет оптимизировать не только операционную деятельность, в частности оптимизацию доставки заказов, но и актуально при проектировании и реорганизации логистических сетей компаний. Вышеуказанные алгоритмы решения ЗМТ не требуют большого количества оперативной памяти для вычислений. В задачах, использующих матрицу стоимости переездов, память в основном расходуется на хранение данных о затратах на перемещение транспортных средств.

ЛИТЕРАТУРА

1. Clarke G. Scheduling of vehicles from a central depot to a number of delivery points / G. Clarke, J.W. Wright // Operations Research. — 1964. — № 12 — P. 568-581.
2. Vigo D. A heuristic algorithm for the asymmetric capacitated vehicle routing problem // European Journal of Operational Research. — 1996. — № 89. — P. 108-126.
3. Gaskell T.J. Bases for vehicle fleet scheduling // Operational Research Quarterly. - 1967. - № 18. - P. 281-295.
4. Yellow P. A computational modification to the savings method of vehicle scheduling // Operational Research Quarterly. — 1970. — № 21. — P. 281-283.
5. Golden B.L. Implementing vehicle routing algorithms / B.L. Golden, T.L. Magnanti, H.Q. Nguyen // Networks. — 1977. — № 7. — P. 113-148.
6. Paessens H. The savings algorithm for the vehicle routing problem // European Journal of Operational Research. — 1988. — № 34. — P. 336-344.
7. Desrochers M. A matching based savings algorithm for the vehicle routing problem / M. Desrochers, T.W. Verhoog // Les Cahiers du GERAD G-89-04, Ecole des Hautes Etudes Commerciales de Montreal, 1989.
8. Altinkemer K. Parallel savings based heuristic for the delivery problem / K. Altinkemer, B. Gavish // Operations Research. — 1991. — № 39. — P. 456-469.
9. Lin S. Computer solutions of the traveling salesman problem // Bell System Technical Journal. — 1965. - № 44. - P. 2245-2269.
10. Christofides INT The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, C. Sandi, editors/ N. Christofides, A. Mingozzi, P. Toth // Combinatorial Optimization. — Wiley, Chichester, 1979. — P. 315-338.
11. Gillett B.E. A heuristic algorithm for the vehicle dispatch problem / B.E. Gillett, L.R. Miller // Operations Research. — 1974. - № 22. - P. 340-349.
12. Wren A. Computers in Transport Planning and Operation. — Ian Allan, London, 1971.
13. Wren A! Computer scheduling of vehicles from one or more depots to a number of delivery points / A. Wren and A. Holliday // Operational Research Quarterly. - 1972. - № 23. - P. 333-344.
14. Fisher M.L. A generalized assignment heuristic for vehicle routing / M.L. Fisher, R. Jaikumar // Networks. - 1981. — № 11. - P. 109-124.
15. Bramel J.B. A location based heuristic for general routing problems / J.B. Bramel, D. Simchi-Levi // Operations Research. — 1995. — № 43. — P. 649-660.
16. Renaud J. An improved petal heuristic for the vehicle routing problem / J. Renaud, F. F. Bostor, G. Laporte // Journal of Operational Research Society - 1996. — № 47. - P. 329-336.
17. Renaud J. A fast composite heuristic for the symmetric traveling salesman problem / J. Renaud, F.F. Boctor, G. Laporte // INFORMS Journal on Computing. - 1996. - № 8. - P. 134-143.